**📘 What is File Upload in Web Apps?**

File upload allows users to send files (like images, documents, etc.) from their computer to the server via a web form.

⚠️ In Node.js, express doesn't handle multipart/form-data (used for file upload) by default. So we use multer middleware.

**🔧 What is multer?**

multer is a middleware for Express.js used to handle file uploads.

* Parses multipart/form-data
* Saves uploaded files to your server
* Supports file filtering and naming

**✅ Why do we need file upload?**

**🧑‍💼 Real-life examples:**

* 👤 Profile picture upload (LinkedIn, Facebook)
* 📄 Resume upload in job portals
* 🖼️ Upload images in blogs or e-commerce sites
* 📧 Attachments in email clients

**📦 Installing multer**

bash

CopyEdit

npm install multer

**📁 Basic Setup: Upload a single file**

**🔨 Backend (Express + multer)**

js

CopyEdit

const express = require("express");

const multer = require("multer");

const path = require("path");

const app = express();

// Storage engine config

const storage = multer.diskStorage({

destination: (req, file, cb) => {

cb(null, "uploads/"); // folder to store uploaded files

},

filename: (req, file, cb) => {

const uniqueName = Date.now() + path.extname(file.originalname);

cb(null, file.fieldname + "-" + uniqueName); // eg: profilePic-16231.png

},

});

const upload = multer({ storage: storage });

app.post("/upload", upload.single("profilePic"), (req, res) => {

console.log("Uploaded file:", req.file);

res.send("File uploaded successfully");

});

app.listen(7100, () => {

console.log("Server running on port 7100");

});

**🧪 Frontend (using React or HTML)**

html

CopyEdit

<form action="http://localhost:7100/upload" method="POST" enctype="multipart/form-data">

<input type="file" name="profilePic" />

<button type="submit">Upload</button>

</form>

**🔄 upload.single(fieldName) vs upload.array(fieldName) vs upload.fields()**

| **Method** | **Use case** |
| --- | --- |
| upload.single("photo") | Upload one file |
| upload.array("photos", 5) | Upload multiple files under one key |
| upload.fields([{ name: "doc" }, { name: "image" }]) | Upload files from multiple fields |

**🧠 How multer works behind the scenes**

1. Parses the form data
2. Extracts files from the request body
3. Stores them in the location you configured
4. Adds a req.file or req.files object

**🔐 Security & Best Practices**

* ✅ Limit file size using limits property in multer
* ✅ Validate file types (images, PDFs only etc.)
* ✅ Store only on secure servers or cloud (S3, Cloudinary)
* ✅ Never trust file names from user input

**File type filter example:**

js

CopyEdit

const fileFilter = (req, file, cb) => {

if (file.mimetype.startsWith("image/")) {

cb(null, true);

} else {

cb(new Error("Only image files allowed!"), false);

}

};

const upload = multer({ storage, fileFilter });

**💻 Real-World Use Case: Profile Picture Upload Flow**

**🔄 Flow:**

1. User selects file → uploads via form
2. Frontend sends multipart/form-data request
3. Backend saves file with multer
4. File is stored locally or uploaded to cloud
5. File path stored in DB for future reference

**📂 Folder structure best practice**

bash

CopyEdit

/project-root

/uploads <- store files (local dev only)

/routes

/controllers

/utils

**🧪 Bonus: Upload to Cloud (like Cloudinary or S3)**

* Use multer with cloud storage adapters
* Great for production apps (scalability + security)

**🧼 Cleanup: Deleting uploaded files**

You can delete files using fs.unlinkSync(path) when:

* User cancels account
* New image replaces old one

**✅ Summary**

| **Concept** | **Explanation** |
| --- | --- |
| multer | Middleware for file upload in Express |
| .single() / .array() | To accept one or more files |
| Storage config | Decide where and how files are saved |
| File filters | Restrict file types |
| Frontend | Must send multipart/form-data |
| Real Use | Upload profile pictures, documents, etc. |